Amendments to the Specification:

Please replace the paragraph beginning at page 1, line 13 and continuing to line 15, with the following rewritten paragraph:

-- This application is a divisional application of copending application serial number 09/671,304, filed September 28, 2000 (now U.S. Patent No. 6,741,983) and which claims priority from provisional U.S. application serial numbers 60/156,452, filed September 28, 1999, which is hereby incorporated by reference in its entirety. -

Please add the following two paragraphs after the paragraph beginning and ending at page 7, line 19 as follows:

--Figure 29 represents a method of indexed storage and retrieval of multidimensional information according to a first embodiment.

Figure 30 represents a method of indexed storage and retrieval of multidimensional information according to a second embodiment.--

Please replace the paragraph beginning at page 8, lines 16-28 and continuing to page 9, line 1, with the following:

--A "tree" is a directed graph that satisfies two properties: (1) for any two nodes n_1 and n_2, a path exists from node n_1 to node n_2, or a path exists from node n_2 to node n_1, (the graph is connected); and (2) no two nodes n_1 and n_2 exist for which paths exist from node n_1 to node n_2 and from node n_2 to node n_1 (the graph is acyclic). For purposes of the invention, a tree can be either a directed graph or an undirected graph. The "root" or "root node" is the unique node of a tree that is not a terminal node for any path in the tree. A "non-terminal" or "non-terminal node" is a node of a tree that is an originating node for at least one path in the tree; (Fig. 30, 3030 and 3040). A "leaf" or "leaf node" is a node of a tree that is not a non-terminal node. A "subtree" of a tree (N, B) is defined uniquely by any node n_r of the tree, and is the tree (N_s, B_s) formed of the set N_s containing the node n_r and all nodes $n \in N$ that are reachable from node n_r, and the set B_s containing all branches that are in paths in the tree that originate at node n_r. Node n_r is the root node of the subtree (N_s, B_s). As referred to herein, a "node" can be a carrier of information or data.--

Please replace the paragraph beginning at page 14, line 1 and ending at line 20, with the following:

--A key to implementation of the search specification on a tree-structured database is what occurs at the nodes of the database tree. These nodes can be C++ objects and can contain partition objects used to describe how the database is segmented at each node. Two types of partitioning at the nodes are illustrated: entropy-adjacency partition assignment (See Fig. 29) and data clustering using multivariate statistical analysis (See Fig. 30). The database is implemented using a Search Queue and one or more Search Engines in each computer host in a single or parallel computer environment. The Search Queue holds Search Requests and additional information such as requests to store or delete DNA profile information in the database. The Search Engines take elements from the Search Queues and perform the requested activities. During this process, additional Search Requests may be generated, which each Search Engine places in the Search Queue. The CODIS search engine communicates with clients that request service across a network interface, and returns the requested information to these clients. This process is shown schematically in Figure 3 for the single host case. Multiple hosts in a parallel computing environment are accommodated using multiple communicating copies of this process. The hosts can either all operate on the same database, or each can contain a portion of the database; a mixture of the two methods can also be used. As an example, communicating groups of processors may operate where all members of each group are assigned the same portion of the database.--

Please replace the paragraph beginning at page 16, line 7 and continuing to line 19, with the following:

--To minimize worst-case search time, division of the database into N roughly equal portions at each level of the database tree is highly desirable. A simple and fast test is needed to accomplish this. One test method that can be used to accomplish this is entropy-adjacency partition assignment; refer to Fig. 29, (e.g., 2910). This method assigns members of the set of possible allele pairs at a specified locus to groups. The goal is to choose these groups so that their expected sizes are roughly equal, and so that alleles with indices that differ by a small number (corresponding to the number of repeated sequences for STR DNA profiles and the number of base pairs for RFLP DNA profiles) have a high probability of being assigned to the same group. By preferentially assigning alleles differing by a small number of base pairs to the same group, the growth of the number of generated search requests due to a client's specification of equivalent alleles will be less than would be the case for other assignments.--

Please replace the paragraphs beginning at page 17, lines 3-30, continuing with page 18, lines 1-30 and ending at page 19, line 3 with the following:

--The partition assignment problem (refers to Step 2920; Fig. 29) can be solved by a global optimization procedure based upon simulated annealing. In this method, an initial random assignment is chosen, and its cost (entropy) is calculated. The assignments are represented by non-negative integers. In the figure, a division of the allele pairs into six partition members is desired, and the members are labeled with the integers 0 through 5, inclusive. The optimization procedure randomly chooses an allele pair to modify and randomly chooses a proposed new assignment for that pair. The change in cost that would result if the new assignment were used is calculated, and if the cost decreases the proposed change is accepted. If the cost increases the proposed change is accepted with a probability p that begins with unity and declines monotonically with iteration number as the optimization process proceeds. An exponentially decreasing probability of acceptance (a geometric sequence) has been found to work well in practice. The optimization procedure terminates when either the cost has not been further decreased over a specified number of iterations or a maximum number of iterations has been achieved. The last computed assignment is used as the solution to the problem. A variation of this procedure, which is used in the Examples, is to maintain a copy of the best (lowest cost) assignment achieved through the current iteration, updating this as better assignments are found, and to use the last best assignment as the optimal assignment.

Preferential assignment of adjoining allele pairs can be achieved by introducing a cost associated with the absence of adjacency. For every allele pair (A), the four (less on the boundaries) assignments for allele pairs that differ by one index in one allele are examined, and a count variable is initialized to zero. For every assignment that differs from the assignment of allele pair (A), a one is added to the count variable. The count variable is then scaled by the probability of the allele pair's occurrence, and these scaled values are summed over all possible allele pairs to form the adjacency cost. An allele pair with zero probability of occurrence can allow the assignment of that pair to be arbitrarily made without regard to adjacency. To avoid this problem, a small number can be added to the probabilities of occurrence, or to those that are zero, causing the assignment to affect the cost. The results reported herein utilized a value of 0.005 added to all allele pair probabilities of occurrence. The resulting adjacency cost is linearly combined with the entropy cost, and the combined cost is minimized using the global optimization procedure. This can be expressed by the equation

$$\text{Cost} = \text{Entropy} + \text{Weight} * \text{Adjacency}$$

where Entropy is the cost due to the non-uniform size of the partition members, Adjacency is the cost due to the existence of adjacent allele pairs having different assignments, and Weight is a non-negative number defining the relative importance assigned to the adjacency cost; see step 2910, Fig. 29, for example.

For certain linear combinations, this cost function results in adjacent groups of allele pairs being assigned to the same partition member without drastically impacting the entropy (measure of non-uniform partitioning) of the result. This effect can be seen visually in Figure 5, where the allele pairs of locus d15s539 are partitioned into six groups. For this partition assignment the entropy is 1.01201, whereas for the assignment shown in Figure 4 the entropy is 1.0111.

This process can be carried to an extreme by weighting the adjacency cost too heavily. In this case, the number of partition members decreases with some members containing zero elements. This effect is visible in Figure 6. At the present time a precise way to select the "best" trade-off between entropy and adjacency is not known. If minimization of entropy cost is too heavily favored, search performance using equivalence and RFLP error tolerances will be adversely affected. If adjacency is too heavily favored, the database tree will be become unbalanced, resulting in "long legs" and poor worst-case performance. “Engineering judgment" can be used to select a partition map (via the weight) from many computed solutions that will yield acceptable performance. This can be done by computing optimal solutions to the assignment problem (step 2930, Fig. 29) for a variety of non-negative weights. If the weight is too large not all partitions will contain assigned allele pairs. If the weight is too small assignments similar to those shown in Figure 4 will be observed. Iteration may be required to determine suitable values.--

Please replace the paragraphs beginning at page 19, line 7 and ending at line 25, with the following:

--A schematic representation of the database tree was presented in Figure 2. In that figure, each node of the tree is represented as being implemented using an entropy-adjacency partition. In practice, this is only one of two methods that may be used at a node, and the tree may contain a mixture of the two cases. The implementation of the tree nodes using entropy-adjacency partitions (Fig. 29) will be discussed in detail in this section; however, the implementation of the tree nodes can also be accomplished using data clustering (Fig. 30).

A decision tree node can be implemented by a C++ Node object, as shown schematically in Figure 7. The object can contain a unique identifying integer stored in the thisnode parameter. A Node object may be either a leaf or non-leaf (non-terminal) tree node, as specified by the Node data element isleaf (Fig. 30, 3040). If it is a leaf, the node can store DNA profile information located at that portion of the tree in a storage data structure. As DNA profiles are being stored into the database, a threshold is utilized to determine at what point a leaf node should be converted to a non-terminal node, resulting in two or more nodes one level below the node (Fig. 30, 3040). Nodes can track the total number of DNA profiles they reference in the nmembers parameter. The offset parameter can be used when stored DNA profiles are located out of processor memory, for example on a disk drive to locate the information relative to the start of the storage media.--

Please replace the paragraphs beginning at page 23, lines 26-30, and continuing through page 24, lines 1-16, with the following:

EXAMPLE 5

Multivariate Statistical Clustering Method (Fig. 29, 30)

This section provides a description of a method that uses multivariate statistical methods to determine clusters (Fig. 30, 3010) that can be utilized to partition portions of a database into groups of roughly equal size (Fig. 29, 2920). As a result, this method generates partition information that can be incorporated within or associated with an arbitrary Node object in a database tree; (Fig. 30, 3040). The application of this method to DNA profile data based upon amplification of short tandem repeat (STR) DNA locus data is presented. For this case, a clear binary encoding of the alleles present at a STR locus is available. For other data types, such as DNA RFLP allele (band) data, the proper choice of a binary encoding scheme is not as easy to determine, and at the present time the binary encoding is necessary.

The DNA STR profiles in the database are first represented in a binary form, using a '1' to denote the presence, and a '0' to denote the absence of an allele at a locus. Based on the allele distribution patterns among two chosen STR loci, clusterable patterns are discernable after principal component analysis of the data matrix. Distinct clusters, usually less than 10, can be established using a clustering method, such as k-means [12]; (Fig. 30, 3010). The membership of each cluster is then identified and recorded. Each DNA STR profile belongs to one and only one of these clusters. Thus, the entire set of DNA profiles in the data base can be partitioned into these clusters, based on the allele distribution at these two chosen loci.--

Please replace the paragraph beginning at page 25, line 11 and ending at line 22, with the following:

--EXAMPLE 7

Clustering by Principal Component Analysis (Fig. 30, 3010)

Principal component analysis (PCA), a popular method of performing multivariate statistical analysis, represents a matrix of high dimension, consisting of correlated information, with a much lower dimensional matrix, without sacrificing significant information contained in the original data matrix. PCA involves a rotation from the original frame of reference to a new frame of reference, whose axes are given by the principal components from the PCA. The first principal component represents the direction along which the variance exhibited by the original data points is maximized. The second principal component, orthogonal to the first, represents the direction along which the remaining variance is maximized. Additional principal components are defined in a similar fashion.--

Please replace the paragraph beginning at page 28, line 12 and ending at line 20, with the following:

--EXAMPLE 11

Identification of Clusters

A clustering algorithm can be employed to perform clustering of the scores projected onto a 2-d principal component space; Fig. 30, 3010). K-means [12] is selected because of its wide use and simplicity. However, with k-means the number of clusters has to be specified before the algorithm can begin. This is not a problem because the choice of the two loci, the two principal components on which to project the data, as well as the number of clusters associated with the scores, are all identified by a priori visual inspection and recorded.--

Please replace the paragraph beginning at page 32, line 1 and ending at line 4, with the following:

--**Table 2. Combinations of two loci that give good clusters**. The corresponding principal components are shown, as well as the number of clusters. V1 and V2 denote the identity of the first and the second principal component specification, onto which good clustering of the projected scores is observed; (Fig. 30, 3010).--

Please replace the paragraph beginning at page 33, line 1 and ending at line 19, with the following:

--EXAMPLE 16

Cluster Formation

The allele frequency distributions for the 2-loci combinations that yielded good clusters were examined to discover the reason behind their clusterability. It was found that those loci with allele probability concentrated at just a few alleles (2 to 4) are good candidates to give good clusters. The main reason is that with just a few alleles possible, the joint 2-loci allele distribution tends to concentrate in those allele pairs with relatively high probability of occurrence; (Fig. 30, 3010). Thus less but more distinct clusters tend to be formed. Figures 14 and 15 show the relative frequency of occurrence of the alleles at the d13s17 and d16s539 locus, respectively. Notice that alleles 11 and 12 in both loci have a much higher probability of occurrence. Figures 16 and 17 show the joint 2-allele frequency distribution for the d13s17 and d16s539 locus respectively. It is noted that only a few of the allele pairs have relatively high probability of occurring. This distribution pattern is to be contrasted with one where the majority of the allele have some probability of occurring but none is much higher than others. Figure 18 shows the joint 2-loci allele-pair probability density for the d13s16 and d16s539 loci. Again, it is observed that most probability densities are concentrated at a few selected allele pairs, corresponding to those alleles with relatively high probability of occurring within each locus.--

Please replace the paragraphs beginning at page 35, lines 14-30, and continuing through page 36, lines 1-25, with the following:

--EXAMPLE 19

Clustering by k-Means and Differentiation of Clusters

The nine distinct clusters can be established analytically by the k-means cluster algorithm. Clusters identified by k-means were validated by visual inspection. Memberships within each cluster were analyzed to get at the similarity among the members; (Fig. 30, 3010). Figure 21 shows a plot of the fraction within each cluster possessing each allele. It is observed that clusters differ in the combination of alleles at each of the 2 loci that are dominant (allele 11 and 12 of both loci). For instance, members of cluster 1 all have the 5th allele of the d13s17 locus (allele 11) and the 8th allele(17-9=8; d13s17 has 9 alleles) of the d16s539 locus (allele 12). From the make up of the principal components, the projections of each clusters onto each principal component can be predicted by looking at the presence or absence of these alleles in the members of the clusters.

Because the most probable alleles for the d13s locus are alleles 11 and 12, and the most probable alleles for d16s539 are alleles 11 and 12 (or index number 16 and 17 in Figure 21 below), the clusters correspond to profiles with various combinations of presence or absence of these dominant alleles at these four positions. Table 3 shows the combinations of these four dominant alleles in each of the nine clusters, based on the plots shown in Figure 21. The assignment of the allele distribution in these four dominant alleles in each of these nine clusters as well as the factor that caused the points to cluster this way is further elaborated below.

From Table 3, Boolean expressions can be written that form logical tests on the data to determine cluster assignment; (Fig. 30, 3010 and 3020). For example, a Boolean expression testing for membership of a DNA profile in cluster 1 is "(d13s17-allele11) and not (d13s17-allele12) and not (d~~13~~16s539-allele11) and (d~~13~~16s~~17~~539-allele12)", where the terms in parentheses are logical variables that are true if the corresponding allele is present and false otherwise. A more complex example is the Boolean expression testing for membership in cluster 5: "(((d13s17-allele11) and (d13s17-allele12)) or not ((d13s17-allele11) or (d13s17-allele12))) and (((d16s539-allele11) and (d16s539-allele12)) or not ((d16s539-allele11) or (d16s539-allele12)))". This expression requires both alleles from each locus to be either present or absent in order to be true. Boolean expressions can be rewritten in various forms and simplified according to methods that are well known and practiced in the fields of Boolean algebra and logic circuit design; (Fig. 30, 3030 and 3040).

Table 3 can also be utilized to form a decision tree that sequentially applies tests for the presence or absence of alleles at specific loci using the methods of inductive inference that were pioneered by J. Ross Quinlan [13] and are well known and practiced in the fields of computer science and engineering. In this case, each node of the database tree that utilizes clusters derived from the multivariate statistical analysis method would contain a decision tree specifying the sequence of tests to be applied to DNA profile targets at that node, and the database tree can be rewritten by expanding these nodes and incorporating the decision tree's nodes into the database tree; (Fig. 30, 3030).--

Please replace the paragraph beginning at page 37, line 1 and ending at line 2, with the following:

--Table 3. The presence (1) or absence (0) of alleles 11 and 12 in the d13s and d16s loci for each scores cluster as shown in Figure 13; (Fig. 30, 3040).--

Please replace the paragraph beginning at page 44, lines 1-29, and continuing to page 45, lines 1-2, with the following:

--EXAMPLE 22

Clustering Analysis of a Real DNA STR Data Set

All the work reported above was done with 10,000 synthetic data profiles, generated based on the allele frequency distribution data for Caucasians as given in the CODIS data base. Recently, Budowle [10] released the STR profiles of six ethnic groups, each of which has around 200 samples. We tested whether PCA clusters from these data would project to the same clusters as that of the synthetic data, using the principal components from the latter to do the projection, and if the relative sizes of the clusters were maintained; (Fig. 30, 3010 and 3020). Therefore, a small Caucasian sample data set from one of the six real DNA sample set was chosen for further analysis. This was done to determine whether or not new data inserted in the database would tend to degrade the balanced structure of the database tree and thus adversely affect mean and worst case search times.

The sample set was first converted to the binary representation format, with 1's and 0's. The corresponding allele information in the d13s17 and d16s539 loci was extracted. This was followed by computing the scores matrix onto the 2nd and 3rd principal components of the large synthetic data set. Figure 23 and 24 show the results. In Figure 23, the scores points from the large data set are overlaid on top of the scores points from the small data set. Figure 24 shows the same thing except in this plot, the scores from the small sample set are overlaid on top of those of the large sample set. The black points depict the scores from the small sample set. Since there are only 176 of them, they do not completely cover the 10000 score points from the large data set. It is evident that the plotted scores from the small data (which are mapped to the same 2-d coordinates in a cluster as the scores from the large data set and are plotted as darker dots) are completely covered by those of the large data set (the dark gray points). This was interpreted to mean that there is no profile present in the small real DNA sample set that is not present in the large synthetic data set. This complete coverage is not always the case. Studies using other 2-loci combinations sometimes yield incomplete coverage. In all instances studied to date, however, the plotted 2-d coordinates for points in the small datasets were easily associated with clusters identified using the synthetic data set; (Fig. 30, 3020). --

Please replace the paragraph beginning at page 46, line 5 and ending at line 16, with the following:

--It is firmly established that DNA STR profiles can be partitioned into distinct clusters using the PCA approach. The partition is based on the allele distribution pattern at 2 loci. Certain 2-loci choices yield much better clustering than others. The factors that determine good clustering and the reason for the clustering have been presented and discussed. Successive partitioning using a different 2-loci combination approach at each round will reduce very quickly the members present within each resultant cluster. Partitioning by PCA clustering can be inserted into a suitably chosen non-terminal Node object of the database tree structure, for searching for matching profiles against a target profile; (Fig. 30, 3040). After passing through this node, it is expected that the number of candidate profiles for search will be reduced by approximately one order of magnitude. (Seven to nine clusters usually result from PCA clustering in which the clusters are about equal in size.)--